Cloud Native Data Platform for Network Telemetry and Analytics

Daniel Tovarňák
Masaryk University
Brno, Czech Republic
tovarnak@ics.muni.cz

Matúš Raček
Masaryk University
Brno, Czech Republic
racek@ics.muni.cz

Petr Velan
Masaryk University
Brno, Czech Republic
velan@ics.muni.cz

***Abstract*—In this manuscript, we present a prototype of a modular data platform that is able to continuously ingest, process, retain, and analyse large amounts of network telemetry data in a scalable and straightforward manner. It follows a recently proposed Data Lakehouse architectural pattern, which is an evolution of two well-known approaches used in this area – data warehouses and data lakes. The platform is based on open standards and open-source components, and it follows cloud native principles in order to be able to run in modern computing environments such as public, private, and hybrid clouds. The primary focus of the prototype is network telemetry and analytics over traffic flows and infrastructure logs for the purposes of cyber-security digital forensics and incident response. During the demonstration part, we will further describe internal workings of the presented data platform and showcase its capabilities and possible applications on a public dataset.**

***Index Terms*—Data Lakehouse, Network Flows, Log Data**

I. Introduction

Modern networks and computing environments produce vast amount of telemetry data that are used for multitude of purposes in organizations of all sizes. Having visibility of the managed network is essential for a wide spectrum of mission-critical jobs, e.g. accounting, capacity planning, performance monitoring, incident handling, maintenance, or fault-detection. The ability to continuously ingest, process, retain and analyse network telemetry data is also a prerequisite for effective automation of IT operations processes. We recognize two large classes of network telemetry data that are very different in their nature.

The foundational concept of *network traffic flows* was proposed in 1991 to facilitate accounting of network usage. Cisco's NetFlow soon became a de-facto standard for acquiring traffic telemetry data, and after millennium, it started to be used for anomaly detection and traffic analysis [1]. Nowadays, the most common tools for flow processing (e.g. *nfdump*) are no longer sufficient for the vast amounts of data exported from modern networks. The user requirements have also changed, and data science approaches are desirable. Even though network flows are essentially incorruptible and they are very accurate, they can lack application-level information, due to encryption.

Infrastructure logs, on the other hand, are considered to be one of the few mechanisms available for gaining visibility into the respective elements of the network and the computing environment itself [2]. Regardless if it is a hardware appliance, router, switch, firewall, IDS/IPS, operating system, web server, or DNS server, the generated developer logs, audit logs, instrumentation data, and other metrics are all an invaluable and unique source of information. When combined, they too represent a high-velocity, high-volume flood of data.

For over a decade, the application domain of network telemetry and analytics has been one of the research and innovation drivers in the area of big-data management. Historically speaking, due to the data explosion and the increase of different applications, the traditional approaches based on *data warehouses* and ETL jobs started to be increasingly complex and costly. Therefore, new highly-scalable and cost-effective concepts gradually emerged. One of the most prominent ones, *data lakes*, have introduced their own set of challenges related to data curation, metadata management, and lack of transactional access, known from relational databases [3]. In recent years, an ever-increasing demand for high-quality structured data access, e.g. driven by machine-learning applications, has supported the emergence of novel approaches and technologies that nowadays allow data engineers to make yet another paradigm shift.

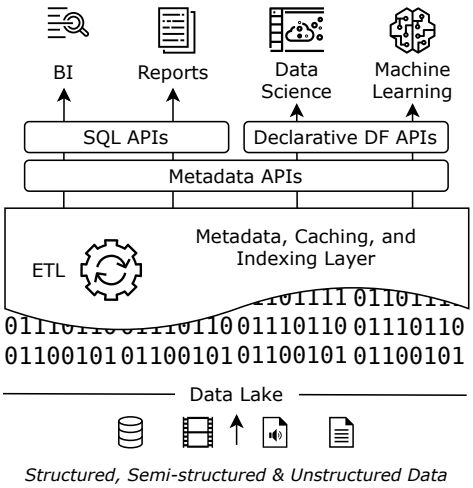


Figure 1: Data Lakehouse storage model [4]

As described in [4], Data Lakehouse is a *data management system based on low-cost and directly-accessible storage that also provides traditional analytical DBMS management and performance features such as ACID transactions, data*

versioning, auditing, indexing, caching, and query optimization. Lakehouses thus combine the key benefits of data lakes and data warehouses: low-cost storage in an open format accessible by a variety of systems from the former, and powerful management and optimization features from the latter.

In architectural terms (see Figure 1), Data Lakehouse can be viewed as a combination of a cheap data lake storage system with a transactional metadata layer on top, which is responsible for transaction management, data versioning, auditing, indexing, caching, query optimization, and schema enforcement. The data are continuously normalized, transformed, and curated via batch and streaming ETL/ELT jobs. Structured access to data can be achieved via distributed SQL query engines, or declarative jobs executed by distributed processing engines.

II. Data Platform Architecture

Our goal was to design and implement a prototype of an open data platform that would follow the Data Lakehouse architecture, whilst taking the specifics of network telemetry data into consideration. Another objective was to follow the cloud native principles in order to be able to easily deploy the whole platform into the Kubernetes container orchestration system, both in managed and self-hosted variants. Finally, we have focused on interoperability and openness, that is why all the used components are open-source, and both ingestion data formats are de-facto standards. *IPFIX* (RFC7011) is used for network flows telemetry, and *IETF Syslog* (RFC5424) is used for infrastructure logs telemetry. Figure 2 shows the overall architecture of the data platform prototype.

- *IPFIXcol2* is a tool developed by CESNET (Czech NREN) used for collection, processing and storage of IPFIX flow data. In our case, it serves as an ingress point that is able to transform IPFIX data to JSON and write it to Kafka.
- *Syslog-ng* is a high-performance log collector used for parsing, filtering, and rewriting heterogeneous log data. In our case, it serves as an ingress point that is able to transform IETF Syslog data to JSON and write it to Kafka.
- *JSON* is a lightweight, text-based, language-independent data interchange format, which we use for intermediary, semi-structured data exchange.
- *Apache Avro* is a binary serialization format capable of representing complex data structures. During the data serialization and de-serialization, it relies on external data schema to specify its structure. It supports schema evolution and we use it for *in-transit* data.
- *Apache Kafka* is a renowned distributed data streaming platform designed with high throughput in mind. It is responsible for high-volume, low-latency data exchange between the respective components of the prototype.
- *Apache Spark* is a distributed analytics engine for both batch and streaming data. It was developed to remove the limitations of the MapReduce paradigm and extend its usage to other areas such as machine learning or structured streaming. This is where ETL/ELT happens.
- *Apache ORC* is a columnar data storage format developed to improve the performance of MapReduce jobs and reduce the size of the stored data. It also enables the readers to process only the data that are required by the actual query. We use it for data *in-situ*.
- *Apache Iceberg* is an open table format for large analytic datasets saved in a distributed storage. In our prototype, it serves as a transactional metadata layer and it uses several query optimization techniques. Thanks to Iceberg, it is possible to organize ORC data into huge structured tables.
- *Apache Hive Metastore* serves as a metadata store (catalog) for Iceberg. It contains various information about the managed tables, partitions, table columns, and their locations. It allows the prototype to work with the data via SQL. It uses Apache Thrift protocol to communicate.
- *MinIO* is a distributed, high-performance, S3-compatible object storage. The raw ORC/Iceberg data are stored here.
- *Trino* is a distributed SQL query engine able to effectively execute queries over huge amounts of data from multiple sources. It forms an additional layer over Iceberg and it provides further query optimizations and caching.
- *Apache Superset* is an intuitive and easy-to-use open-source platform for structured SQL-based data exploration and visualization. It is used by the users of our prototype to interact with the stored data via web GUI.
- *PostgreSQL* is a well-known RDBMS, which can be used to store relatively small amounts of data (e.g. blocklists) for immediate, low-latency access in SQL queries.

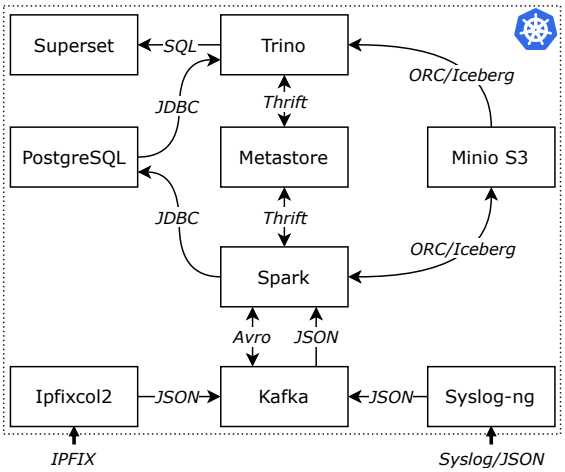
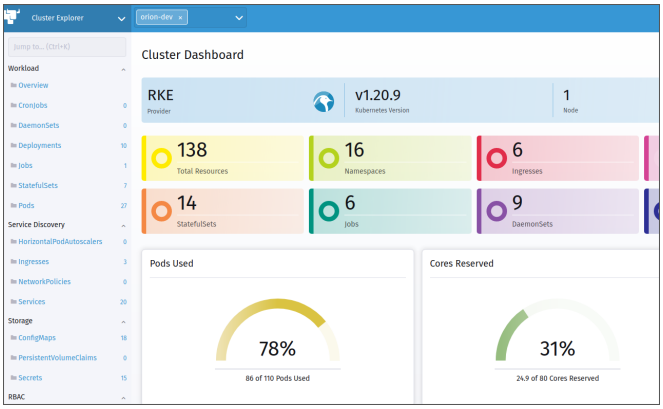


Figure 2: Components of the data platform prototype deployed in Kubernetes

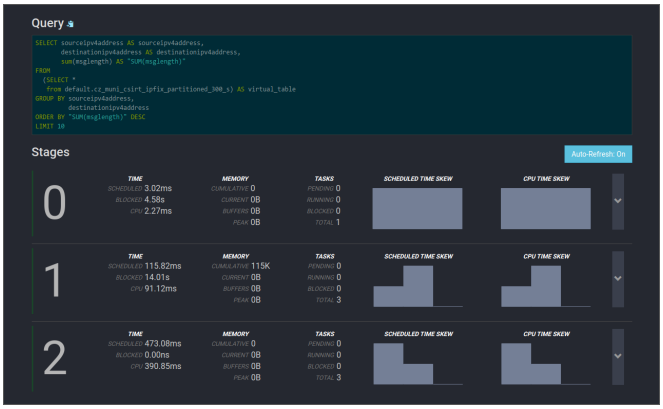
III. Demo Description

In the introductory part of the demonstration, we will first showcase the easy and straightforward deployment of the whole platform into a Kubernetes cluster (see Figure 3a) and discuss the advantages and disadvantages of this approach, e.g. from the perspective of extensibility and scalability.

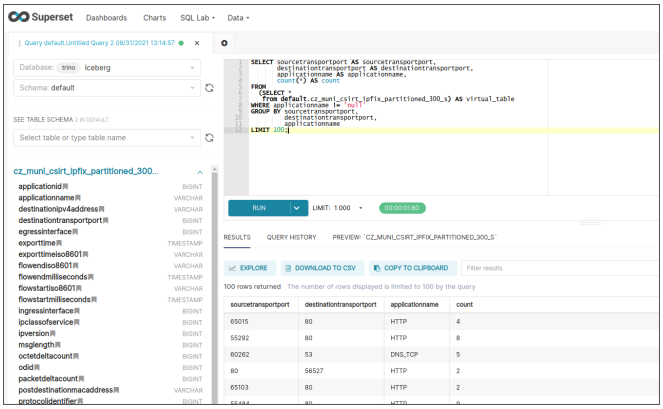
Next, the datasets used throughout the demonstration will be presented. Most prominently, we will use data [5] that were collected from a digital twin of a fictitious organization, hence, they are equal to data generated in real enterprise networks, but, there is no need for their anonymization or obfuscation.



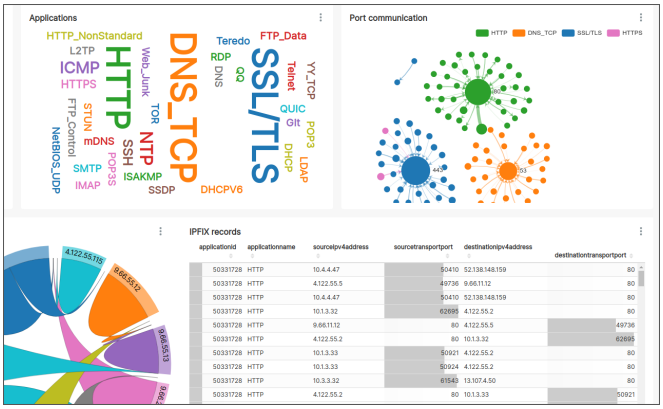
(a) Rancher cluster explorer for Kubernetes



(b) Distributed query overview in Trino



(c) Apache Superset SQL editor



(d) Dashboard example in Apache Superset

Figure 3: Web-based user interfaces for selected components of the presented data platform

Finally, we will describe the full life-cycle of the data as they pass through the platform prototype. First, the raw data are ingested in one of the telemetry formats and converted into an intermediary JSON representation. Alternatively, they can be directly ingested in JSON, with Syslog-ng serving as a proxy. Next, the data are normalized into structured Avro records by their respective Spark streaming jobs. Such records can be further cleaned and transformed in a streaming manner, until they are ready to be stored in Iceberg tables. After storage, the individual records are available to be queried by the Trino SQL engine (see Figure 3b) or to be further processed via ad-hoc Spark batch-processing jobs. At last, the Superset component can be used by the end-users to directly interact with the stored data via web-based UI, as if they were stored in a traditional relational database, but possibly on a petabyte scale.

A particular attention will be dedicated to the possible ways third-parties can interact with the stored data tables, which is essentially the main interest of any end-user. The examples will include ad-hoc batch-processing and streaming jobs for Apache Spark, integration with Python code for automation purposes, and examples of exploratory analysis tasks (see Figure 3c) and visualisation capabilities (see Figure 3d) of Apache Superset.

In the end, real-world applications and experience from daily operations will be discussed, together with possible future directions of our research and activities. These include, but are not limited to, an addition of an SQL-like interface and a processing engine for the execution of real-time streaming queries, and a roadmap of the data platform with respect to its planned open-source release.

Acknowledgements

This research was supported by the Security Research Programme of the Czech Republic 2015–2022 (BV III/1-VS) granted by the Ministry of the Interior of the Czech Republic under No. VI20202022164 Advanced Security Orchestration and Intelligent Threat Management.

References

[1] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[2] A. Oliner and J. Stearley, "What Supercomputers Say: A Study of Five System Logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, 2007, pp. 575–584.

[3] P. Sawadogo and J. Darmont, "On Data Lake Architectures and Metadata Management," *Journal of Intelligent Information Systems*, vol. 56, no. 1, pp. 97–120, Feb 2021.

[4] M. Zaharia, A. Ghodsi, R. Xin, and M. Armbrust, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," in *11th Conference on Innovative Data Systems Research, CIDR 2021, Online Proceedings*, 2021.

[5] D. Tovarňák, S. Špaček, and J. Vykopal, "Traffic and Log Data Captured During a Cyber Defense Exercise," *Data in Brief*, vol. 31, p. 105784, 2020.